ELSEVIER

Journal of Computational and Applied Mathematics 82 (1997) 433–446

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Distributed implicit stepping with unstructured finite volumes for 2D-transport

P. Wilders[a,*], G. Fotia[b]

[a] *Department of Mathematics, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, Netherlands*

[b] *CRS4, Comp. Mech. Group, Via Sauro 10, I-09123 Cagliari, Italy*

Abstract

A detailed description of transport phenomena in real-life complex geometries is important in several areas of reservoir and environmental engineering. This paper presents numerical issues related to the application of high-performance computing in this field. The specific example treated comes from reservoir engineering. A distributed parallel environment is developed based upon domain decomposition. We discuss both the iterative properties and the parallel performance on an IBM-SP2. It will be shown that the numerical approach leads to an attractive tool for treating real-life large-scale problems.

Keywords: Advection–diffusion; Porous medium; Domain decomposition; Distributed computing

AMS classification: 65F10, 76S05, P20

1. Introduction

Transport phenomena play an important role in reservoir and environmental engineering. The physical model consists of an advection–diffusion equations for a concentration combined with flow equations determining the advective velocities of the concentration. The two sets of equations have often quite different characteristics and associated time scales. For this reason, it is of common use to solve practical problems by modeling flow and transport sequentially. This implies that the concentration is transported, assuming fixed flow data during one or multiple time steps. As such, the development of the flow and transport solver might be decoupled. In this paper the emphasis is on transport. We study a 2D transport equation in order to reveal some basic numerical properties of our approach, when applied to real-world problems.

* Corresponding author. E-mail: p.wilders@math.tudelft.nl

PII S0377-0427(97)00052-6

The application dealt within this paper is two-component single-phase miscible flow in a porous medium (reservoir). In this case the flow model is presented by an elliptic equation for the pressure from which Darcy's velocity, i.e. the advective velocity, can be computed. Most often, the IMPES method is applied (implicit pressure explicit saturation), in which an explicit time-integration procedure is used for updating the concentration. However, some stiffness might be present in the transport equation, often due to strong heterogeneity in the reservoirs. This yields severe time-step limitations. There are several ways to overcome this limitation, e.g. by combining explicit and implicit time stepping [19] or explicit high throughput (HT) time stepping [21] or implicit time stepping. In the latter class a very popular method is characteristic backtracking [6], often combined with operator splitting. When dealing with the parallel implementation of this method the communication overhead turns out to be a severe bottleneck; see [12]. Instead, we rely upon a straightforward implementation of an implicit time-advancing scheme (the trapezoidal rule). An important computational issue of any solver based on that approach is the numerical solution of the large nonsymmetrical systems. We present a distributed parallel environment for such a task, based upon domain-decomposition and coarse-grained parallelism. For the automatic grid partitioning we rely upon the public domain package Chaco [9].

The outline of this paper is as follows: In Section 2 we describe the governing equations and underlying physical model. Section 3 gives some details on the spatial discretization. The time-integration scheme and the iterative procedures used for the solution of the nonsymmetrical systems are discussed in Section 4. Section 5 describes some aspects of the parallel implementation and investigates briefly the scalability. Numerical experiments are presented in Section 6. We discuss the iterative procedures and the measured parallel performance characteristics on an IBM-SP2. The problem is concerned with a geometrically irregular reservoir with "real-life" permeability data. Finally, in Section 7 we draw some conclusions and make some remarks concerning future research.

2. The governing equations

We shall consider a scalar conservation law of the form

$$\varphi\frac{\partial c}{\partial t}+\nabla\cdot[\boldsymbol{v}f(c)-D\nabla c]=0,\quad \boldsymbol{x}\in\Omega\subset\mathbb{R}^2,\ t>0, \tag{1}$$

with

$$f(c)=c. \tag{2}$$

The Darcy velocity $\boldsymbol{v}(x,t)$ is obtained from an elliptic equation for the pressure $p(x,t)$:

$$-\nabla\cdot(a\nabla p)=0,\quad \boldsymbol{x}\in\Omega,\ t>0. \tag{3}$$

Darcy's law states that $\boldsymbol{v}=-a\nabla p$.

Eq. (1) is a parabolic equation with an almost-hyperbolic behaviour, due to the dominance of the advective term. The coefficient $\varphi(\boldsymbol{x})$ stands for the porosity of the porous medium and D is a velocity-dependent anisotropic dispersion tensor given by

$$D=\frac{d_\ell}{|\boldsymbol{v}|}\begin{bmatrix} v_1^2 & v_1v_2\\ v_1v_2 & v_2^2\end{bmatrix}+\frac{d_t}{|\boldsymbol{v}|}\begin{bmatrix} v_2^2 & -v_1v_2\\ -v_1v_2 & v_1^2\end{bmatrix}. \tag{4}$$

The parameters d_ℓ and d_t (longitudinal and transversal dispersivity) in the dispersion tensor are small. They can even be negligible, e.g., if the medium presents small capillary effects. In this case, (1) is a pure hyperbolic equation and a typical solution will exibit sharp fronts moving through the porous medium.

As mentioned in Section 1 the sequential or split approach is used for treating the coupling between (1) and (3). This implies that we may assume the velocity to be frozen in (1). Stated otherwise, the coefficients in (1) are assumed to be time-independent (at least within each separate time step).

With the choice (2), a linear flux function, Eqs. (1) and (3) give a model for the miscible displacement of oil by injection of a solvent; see [4, 14]. The unknown c represents the solvent concentration, defined as the ratio of the solvent mass and the total mass (solvent + oil). Miscible transport phenomena in complex porous media occur frequently in practice. For example, enhanced oil recovery (EOR) techniques, which are based upon injection of solvents in reservoirs, make usage of miscible or partial miscible models; see [14]. A second example is in the area of contaminant transport in aquifers; see [13]. It is also possible to capture alternative physical models with (1) and (3): e.g., incompressible immiscible two-phase displacement of oil by waterflooding, a typical secondary recovery technique in reservoir exploitation. In this case it is sufficient to identify the unknown c with the water-phase saturation, defined as the ratio of the volume filled by the injected water and the total porous volume, and to assume a nonconvex flux function of the Buckley–Leverett type. Our numerical methods can be applied to such a case as well with only minor modifications.

For the boundary conditions accompaning (1), we took a prescribed concentration or flux at injection wells, zero Neumann at production wells and zero flux elsewhere. A cold-start, i.e. $c = 0$, was employed as the initial condition in our computations.

The coefficient a in (3) reads

$$a = \frac{k}{\mu}, \tag{5}$$

with $k(\boldsymbol{x})$ the permeability of the porous medium and μ the viscosity. In strongly heterogeneous reservoirs, the permeability might vary by several orders of magnitude and this may lead to quite irregular velocity patterns. For the viscosity the quarter-power viscosity law is often adopted, i.e.,

$$\mu^{-1/4} = (1 - c)\mu(0)^{-1/4} + c\mu(1)^{-1/4}. \tag{6}$$

The time scales at which the Darcy velocity changes are often much larger than the time scales for the concentration; see [5]. Sometimes it is possible to model the solvent as a tracer; see [14, 20]. In this case the viscosity μ is assumed to be a constant and the Darcy velocity is computed only once. Our numerical experiments are concerned with tracer flows. However, the characteristics of the transport solver found for this simplified model will carry over if the sequential interaction between (1) and (3) is elaborated more fully.

3. Space discretization

An advantage of the sequential or split approach for treating the coupling between (1) and (3) is that one can devise effective numerical strategies for (1) and (3) separately, taking proper care of

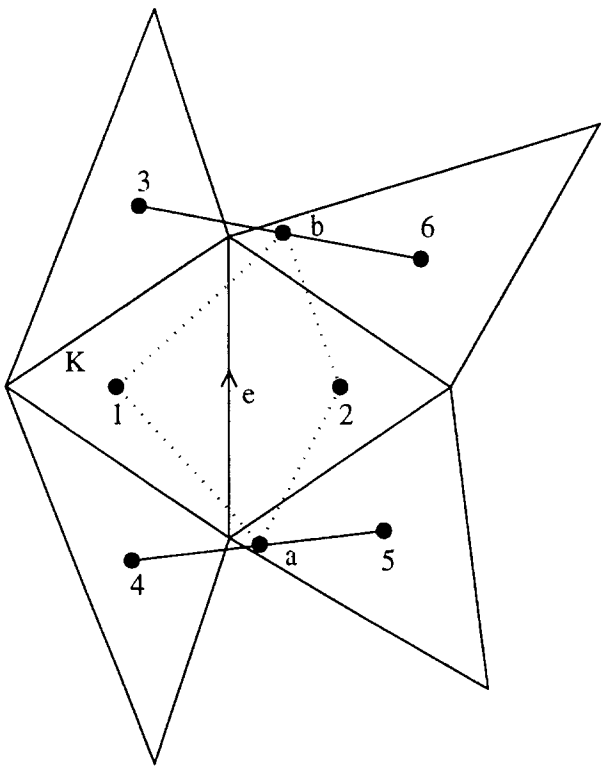

Fig. 1. Edge molecule, shadow volume for gradient.

the different nature of the equations involved. In [7] it has been argued that a convenient numerical approximation can be based upon the triangular mixed finite element method for (3) combined with the cell-centred unstructured finite volume method on the same grid for (1). We use the triangular mixed finite element method of [1] for the discretization of (3) and we refer to this report and cited literature for more details. In the remainder of this section we briefly describe the finite volume approximation of (1).

The domain Ω is covered with a triangular grid with triangles K. The area of the triangle K is denoted by $|K|$, the boundary ∂K is composed of three edges of length $|e|$. Integration of (1) over K, applying Green's formula and inserting a midpoint rule, leads to

$$|K|\varphi_K \frac{\mathrm{d}c_K}{\mathrm{d}t} + \sum_{e\in\partial K} |e| U_{e,K} c_e = \sum_{e\in\partial K} |e| g_{e,K}. \tag{7}$$

Here, φ_K and c_K are centroid values, c_e is and estimate of c on the edge e and $U_{e,K}$ is the normal component of the velocity on the edge e, i.e., $U_{e,K} = \boldsymbol{v}_e \cdot \boldsymbol{n}_{e,K}$ with $\boldsymbol{n}_{e,K}$ the outward unit normal on the edge e. The quantity $g_{e,K}$ is an approximation of the viscous flux $D\nabla c \cdot \boldsymbol{n}_{e,K}$ on the edge. Both c_e and $g_{e,K}$ are evaluated using the edge-molecule depicted in Fig. 1.

For the advective term we adopt a variant of the JST-scheme. The JST-scheme is a central scheme stabilized with a nonlinear artificial dissipation term containing both harmonic and biharmonic

operators. The scheme has originally been developed for the numerical solution of nonlinear hyperbolic equations with emphasis on capturing shock fronts and high gradient internal layers with a moderate level of numerical diffusion. To be more specific we take

$$c_e = \tfrac{1}{2}(c_1 + c_2) - \operatorname{sign}(U_{e,K})[d_e^{(2)} - d_e^{(4)}], \tag{8}$$

$$d_e^{(2)} = \varepsilon_e^{(2)}(c_2 - c_1) \qquad d_e^{(4)} = \varepsilon_e^{(4)}(\nabla^2 c_2 - \nabla^2 c_1), \tag{9}$$

$$\nabla^2 c_K = \sum_{e \in \partial K} (c_{2,e} - c_{1,e}). \tag{10}$$

In the rhs of (10) reference is made to the numbering system in Fig. 1; the extension e in the lower index is to indicate that each edge has its own local numbering system attached to it. The adaptive parameters $\varepsilon_e^{(2)}, \varepsilon_e^{(4)}$ contain nonlinear switch function; see [11, 10]. In [17, 23] the switch functions from the original method have been modified to become differentiable switch functions depending on the gradient of the solution of the edge e. We adopted the latter formulation.

For the approximation of the viscous fluxes we need to express the gradient of c on the edge e in the surrounding centroid values. For this, we use the shadow volume sketched in Fig. 1. Here, we only present the main formula. Details may be found in [23]. The resulting expression for $g_{e,K}$ reads

$$g_{e,K} = [\tilde{e}|(D\tilde{\boldsymbol{n}} \cdot \boldsymbol{n})(c_2 - c_1) + |\bar{e}|(D\bar{\boldsymbol{n}} \cdot \boldsymbol{n})(c_a - c_b)]/\Delta, \tag{11}$$

$$c_a = \tfrac{1}{2}(c_4 + c_5), \qquad c_b = \tfrac{1}{2}(c_6 + c_3), \tag{12}$$

$$\Delta = |\tilde{e}||\bar{e}|\bar{\boldsymbol{t}} \cdot \tilde{\boldsymbol{n}}. \tag{13}$$

The unit normal on the edge e has been denoted before by $\boldsymbol{n}$ (omitting indices) and the length of the edge e by $|e|$. The unit normal on the segment connecting the cell centres P_1 and P_2 is denoted by $\bar{\boldsymbol{n}}$, the unit tangential vector with $\bar{\boldsymbol{t}}$ and the length of this segment by $|\bar{e}|$. Similarly, $\tilde{\boldsymbol{n}}$ is the unit normal on the segment joining the points P_a and P_b and the length is $|\tilde{e}|$. It can be verified that Δ is twice the area of the shadow volume. The approximation (11) is exact for constant and linear functions.

A common approach for the approximation of the viscous fluxes is to connect the cell centres P_1, P_2 with the two vertices of the edge e (see Fig. 1) and to take the resulting region as the shadow volume. Next, the vertex value of the unknown is defined as a weighted average of the surrounding centroid values. In this approach the molecule from Fig. 1 is extended. Our goal was to stay within the molecule of Fig. 1, because we intend to use implicit time stepping with the smallest matrix possible. After summing up the three edge contributions in (7), it can be seen that the final approximation explores the 10-point cell molecule sketched in Fig. 2. It can be easily seen that the constant state $c = 1$ presents a solution of (7) if and only if

$$\sum_{e \in \partial K} |e| U_{e,K} = 0. \tag{14}$$

Relation (14) states that the velocity should be divergence-free in a special discrete sense. In the mixed finite element method the relation (14) automatically holds for triangles from the finite element mesh. Because the same mesh is used for transport the conclusion is that (14) is satisfied. Recently,

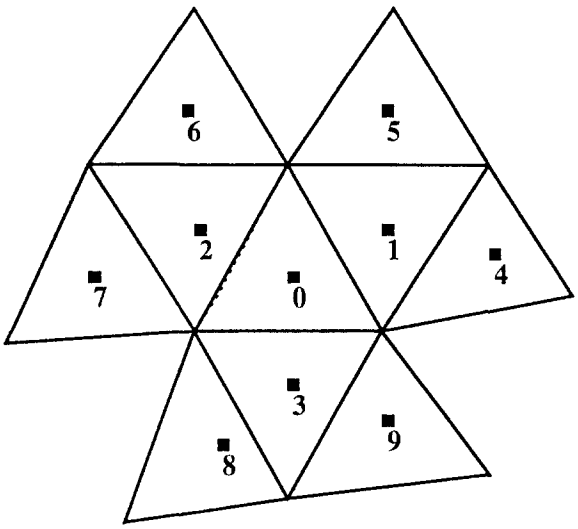

Fig. 2. Cell molecule for cell 0.

we have, in a surface water problem, encountered a case in which different meshes were present and (14) did not hold. It turned out that in such a case the discretization must be modified.

4. Implicit time-stepping and domain decomposition

Taking (7) together for all triangles K, we end up with a system of differential equations for the centroid values, i.e.,

$$M\frac{\mathrm{d}c}{\mathrm{d}t} = Pc. \tag{15}$$

The matrix P is a sparse matrix with entries depending on c as a consequence of the nonlinearity of the artificial dissipation; see (8). The specific formulation of the artificial dissipation is such that the rhs of (15) is differentiable. This enables us to reach the highest level of time accuracy. We set

$$J = \frac{\partial Pc}{\partial c}. \tag{16}$$

The matrix M is a diagonal matrix containing the cell values of the porosity multiplied with the area of the cell. M is fixed and does not depend on time. The entries of the matrix P depend on the coefficients in the spatial operators of (1). As outlined in the Section 2 these coefficients are assumed to be time-independent (at least within each separate time step).

For the time discretization of (1) we use the linearly implicit trapezoidal rule in the increment or delta formulation, i.e.,

$$\left(\frac{1}{\tau_n}M - \frac{1}{2}J^n\right)\delta^n = (Pc)^n, \quad \delta^n = c^{n+1} - c^n. \tag{17}$$

Here, τ_n denotes the time step. The scheme is second-order-accurate in time.

Omitting indices, we denote (17) by

$$A\delta = b. \tag{18}$$

The iterative solution has been based upon a Krylov–Schwarz domain decomposition method. We apply GMRES with minimal overlap additive Schwarz (or block Jacobi) as the preconditioner. Nowadays, this method is well established for elliptic equations; see [16]. It can be shown (see [2]) that an equivalent formulation can be found by applying GMRES to the interface equations

$$Bv = g, \qquad B = Q^{\mathrm{T}}N^{-1}AQ, \qquad g = Q^{\mathrm{T}}N^{-1}g. \tag{19}$$

In the appendix we briefly describe the principles of the proof. Using the interface equations (19) it can also be shown that there is equivalence with a preconditioned Krylov–Schur method; see [22]. In (19), N is the block-Jacobi iteration matrix and Q^{T} the restriction operator from δ to v; v is the vector of interface variables, which are variables living in a small region surrounding the internal boundaries. All triangles of which the molecule (see Fig. 2) crosses the internal boundary are in this region. The formulation in terms of interface equations has some advantages, e.g. the memory requirements put forward by GMRES are minimized and it facilitates the parallel implementation (see Section 5). A disadvantage is that inaccurate subdomain solvers are not allowed, because this would destroy the equivalence property.

In each GMRES ddm-iteration subdomain problems have to be solved. The properties of the subdomain matrices have been studied in [23] and ILU-preconditioned Bi-CGSTAB has been found to be an efficient solver. Here, it is important to remark that the part of the matrix corresponding to the first term on the rhs of (8) is anti-symmetrical because of relation (14).

5. Parallel aspects

The parallel implementation has been based upon coarse-grained parallelism provided by the Krylov–Schwarz domain decomposition method described in Section 4. The computation has been divided into p processes, carrying out $(p+1)$ tasks, numbered $q = 0, \ldots, p$. Each of the tasks $1, \ldots, p$ contain a number of subdomains. Task 0 performs all global and/or sequential operations (e.g., deals with all i/o). Tasks 0 and 1 are taken together to form process 1. For $q > 1$, process q is identical to task q. The processes are the separate units, which are connected via explicit programming and message passing. In our implementation different processes are always assigned to different processors. This means that p processors are doing the job. Furthermore, in this paper we assume that in the parallel run each of the tasks $1, \ldots, p$ contains only one subdomain.

The GMRES ddm-acceleration (see (19)) has been included in task 0. This facilitates the implementation and can easily be done, because of the formulation in terms of interface variables. It is possible to use an existing GMRES-routine after minor modification and to use a simple gather/scatter as the communication scheme. Let us assume that we are in the stage of starting a new GMRES iteration. Then every task $q = 1, \ldots, p$ builds and solves its subdomain problem. The interface variables $v(q)$ residing in its subdomain are passed to task 0 as the (partial) result of the matrix–vector multiplication with the matrix B from (19). Task 0 gathers all $v(q)$, does some sequential operations and updates $v = \{v(q): q = 1, \ldots, p\}$. Next, task 0 scatters the information that is needed in the tasks $1, \ldots, p$ for a subsequent subdomain inversion. In this scheme both the communication time and the

computational time spent in the GMRES-routine depend linearly on M_pV, with M_p the total number of matrix–vector multiplications with the matrix B and V the total number of interface variables. $M_p = O_p + 1$ holds, where O_p denotes the number of ddm-iterations.

For the analysis of parallel aspects we compare the same algorithm applied to the same problem in a parallel and a serial run; to speak in terms of [15], our concern is on parallelizability. In our opinion, an investigation of the parallelizability is an important first step in the parallel analysis. The speed-up S_p and efficiency E_p are defined by

$$S_p = \frac{T_1(p)}{T_p(p)} = \frac{\text{serial elapsed time}}{\text{parallel elapsed time}}, \qquad E_p = \frac{S_p}{p}. \tag{20}$$

Only the computational heart, i.e. the time-stepping loop is taken into account to determine the elapsed time. The lower index in (20) refers to the number of processors. The number in parantheses refers to the problem size; in our case determined by the number of subdomains. Often, this last dependence will be omitted in the notation. In this paper our concern is on the parallel efficiency of the code for solving a real-file problem on a given grid; our concern is on the fixed-size speed-up. For discussions related to scaling methodologies we refer to [8, 18, 3] and cited references.

We have

$$T_p = T^{(0)} + T_p^{(1)} + T_p^{(c)}, \tag{21}$$

with $T^{(0)}$ the computational time spent in task 0, $T_p^{(1)}$ the computational time spent in task 1 and $T_p^{(c)}$ the communication time. Of course, $T_1^{(c)} = 0$. Moreover, task 0 takes the same computational time in the serial and parallel run; we have expressed this by omitting the lower index on $T^{(0)}$ in (21). In the parallel run the tasks $1,\ldots,p$ are fully synchronized. Thus, all tasks $1,\ldots,p$ take the same synchronized time $T_p^{(1)}$. We assume that these tasks are fully load balanced and, because there is architectural scalability on the IBM-SP2, this implies

$$T_p^{(1)} = \frac{1}{p} T_1^{(1)}. \tag{22}$$

Let us define the numerical efficiency $E_p^{(n)}$ by

$$E_p^{(n)} = \frac{T_1(2)}{T_1(p)}. \tag{23}$$

$E_p^{(n)}$ measures the efficiency of the algorithm if the same problem is solved with a varying number of subdomains on a single processor. We took the two subdomain case as the reference case, because this is the lowest number of subdomains for which all the numerics in the code are active. If we solve the problem as a single-domain problem, then the ddm-iteration is switched off completely. Similarly, we define

$$E_p^{(c)} = \frac{T_2^{(c)}(2)}{T_p^{(c)}(p)}, \qquad E_p^{(s)} = \frac{T^{(0)}(2)}{T^{(0)}(p)}. \tag{24}$$

$E_p^{(c)}$ measures the efficiency of the communication and $E_p^{(s)}$ is related to the efficiency of the sequential part. Finally, we set

$$U_p^{(c)} = \frac{T_p^{(c)}(p)}{T_1(p)}, \qquad U_p^{(s)} = \frac{T^{(0)}(p)}{T_1(p)}. \tag{25}$$

Using (22), if follows from (20) and (21) that

$$\frac{1}{E_p} = 1 + pU_p^{(c)} + (p-1)U_p^{(s)}. \tag{26}$$

For scalability, $1/E_p$ must be bounded from above as a function of p. From (26) it is clear that a necessary condition for scalability is that both $U_p^{(c)}$ and $U_p^{(s)}$ tend to zero as $O(1/p)$ for increasing p. Our analysis will show that our code does not meet this condition.

From (23)–(25) it is easy to see that

$$U_p^{(\alpha)} = \frac{E_p^{(n)}}{E_p^{(\alpha)}} U_2^{(\alpha)}, \quad \alpha = c, s. \tag{27}$$

Substitution of (27) into (26) gives

$$\frac{1}{E_p} = 1 + pE_p^{(n)}\left(\frac{1}{E_p^{(c)}}\, U_2^{(c)} + \frac{p-1}{p}\,\frac{1}{E_p^{(s)}}\, U_2^{(s)}\right). \tag{28}$$

It has been argued above that both $T_p^{(c)}(p)$ and $T^{(0)}(p)$ depend linearly on $M_p V$. Both M_p (see later on) and the number of interface variables V will increase if the number of subdomains grows. Thus $E_p^{(c)}$ and $E_p^{(s)}$ are decreasing if p grows. More roughly, there holds

$$E_p^{(c)}, E_p^{(s)} \leqslant 1 \quad \text{for } p \geqslant 2. \tag{29}$$

Combination with (28) proves that there exists a constant K, independent of p, such that

$$\frac{1}{E_p} \geqslant 1 + KpE_p^{(n)}. \tag{30}$$

It shall be clear that $T_1(p)$ primarily depends on the number of ddm-iterations. It is well known (e.g., see [16]), that the number of ddm-iterations is a growing function of the number of subdomains in one-level ddm-methods, of which our Krylov–Schwarz method is an example. This means that $E_p^{(n)}$ is a decreasing function of p. However, as will be established numerically the product pE_p^n is an increasing function of p in practical ranges. The main reason for this is that much work is done in those parts of the code that behave linearly as a function of the number of unknowns, such as building matrices. This means that the numerical efficiency is a rather slowly decreasing function of p.

Finally, we want to comment on the assumption of load balancing leading to (22). If the grid partitioning is load balanced in the sense that every subdomain contains an equal number of grid points, then the only source of load imbalance can be found in a variance of the number of Bi-CGSTAB iterations for solving the subdomain problems. It has been found numerically that some load imbalance is introduced in the code at this point. However, the final influences are small because only a minor part of the work is done here.

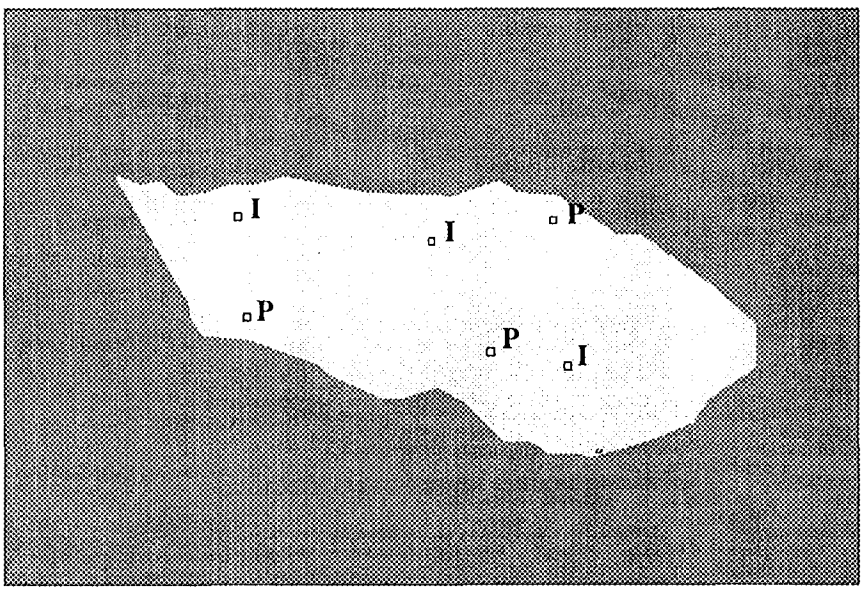

Fig. 3. Geometry of irregular reservoir.

6. Numerical experiments

Our aim in this section is to discuss the parallel performance of the transport solver for a "real-life" problem. The performance of the time integration scheme and, less extensively, the spatial discretization scheme has been a subject of investigation in [23] and we refer to this report for more details. Here, it suffices to mention that the computations described in this section have been done with a maximal Courant number of O(40).

The reservoir under consideration is illustrated in Fig. 3; there are three injection and three production wells. As mentioned in Section 2, our concern is on tracer flows. For the computation of the velocity field we have used strongly heterogeneous realistic permeability data, provided by Agip S.p.A.. Fig. 4 contains a detail of the computed velocity field. A detail of the computed concentration can be found in Fig. 5.

A mesh of 36.500 cells was used. This mesh was divided into p subregions using the public domain package Chaco [9]. The graph given to Chaco was taken from the mesh directly. An alternative would be to take the graph from the matrix. We have asked Chaco for a fully load-balanced partitioning. The analysis, carried out in Section 5, shows that it is vital to minimize the number of interface variables V. V depends linearly on E, the number of edges on the internal boundaries. In Table 1 we present E for three different partitioning methods: I = multilevel with Kernighan–Lin optimization (bisection), II = one-level spectral (bisection), IIa = one-level spectral (bisection) with an approximate eigenvector. Method IIa is (roughly) a factor of 5 more expensive than method I for the mesh under consideration. Method II is a factor of 10 more expensive than method IIa. We also set $\sigma = E/(\sqrt{p} - 1)$ and present σ for method I in Table 1. We see that method I is attractive for the mesh under consideration and we use this method for the partitioning. It is peculiar that method II does not outperform the other methods. On the basis of the analysis of a square domain

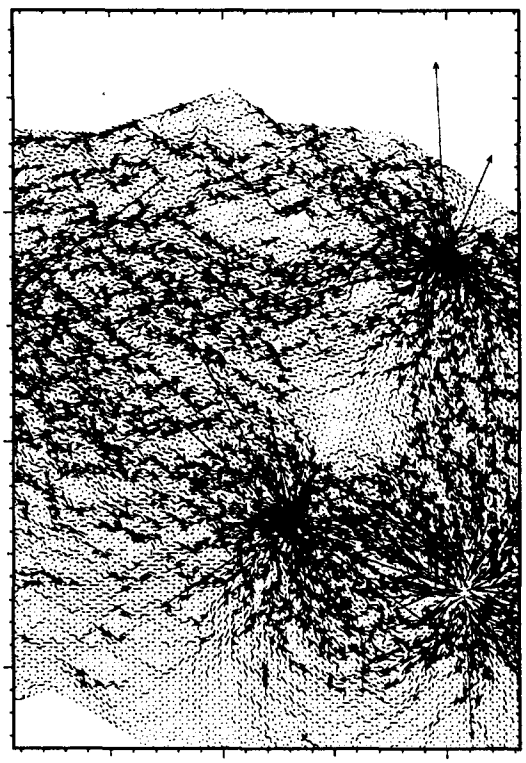

Fig. 4. Detail of the velocity field.

Table 1
The number of edges on internal boundaries

	E			σ
p	I	II	IIa	I
4	298	410	267	298
8	530	563	510	290
16	914	883	847	305
25	1251	1158	1147	313

one may expect E to grow like $(\sqrt{p}-1)$. The values of σ found in Table 1 confirm this hypothesis more or less. Among others, this implies that V increases if p grows, a fact used already in Section 5.

Table 2 contains O_p, the number of ddm-iterations per time step (averaged over the full time interval, i.e. 200 time steps). In Section 5 we argued that O_p increases as a function of p. The expected growth turns out to be quite moderate.

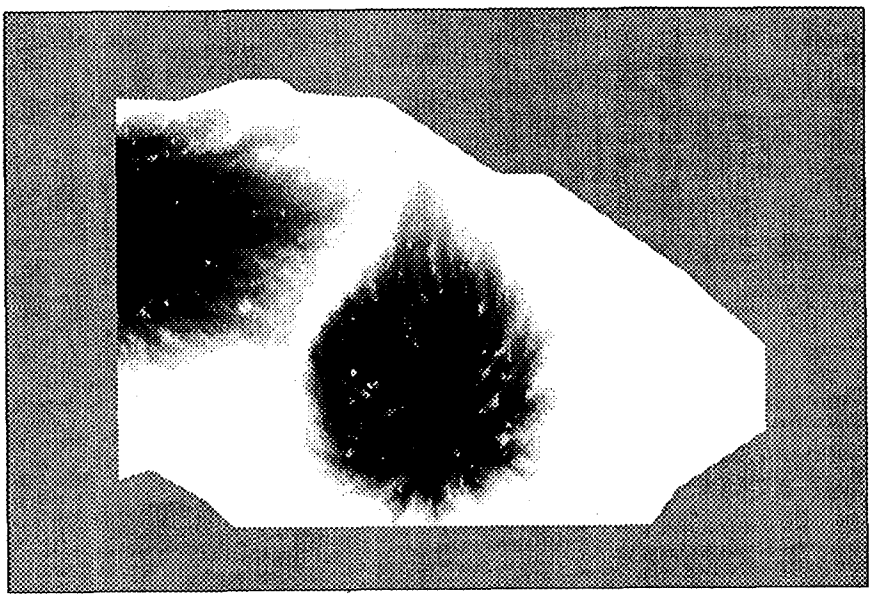

Fig. 5. Detail of concentration.

Table 2
O_p, number of ddm-iterations

p	2	4	8	16
O_p	6.5	8.0	8.5	10.7

Table 3
Speed-up S_p, efficiency E_p and $pE_p^{(n)}$, the factor in the analytic expression of $1/E_p$

p	2	4	8	16
S_p	1.9	3.4	6.0	9.4
E_p	0.95	0.84	0.75	0.59
$pE_p^{(n)}$	2.0	3.6	6.6	10.3

A numerical investigation of the product $pE_p^{(n)}$ was announced in Section 5. Table 3 presents the results. Moreover, Table 3 contains the measured speed-up S_p and efficiency E_p (measured over the full time interval). As expected $pE_p^{(n)}$ grows with p, however, slower than linear. The results in Table 3 are in full qualitative agreement with the analysis of the parallel model in Section 5. A good parallelizability might be observed. It shall be clear that this is closely related to the fact that the number of unknowns in the subdomains is sufficiently high.

7. Conclusions and final remarks

Several features of unstructured finite volumes and implicit time integration for advection–diffusion have been discussed. In particular, the parallel performance for a fixed-size problem was investigated in detail, both analytically and numerically. The numerical experiments were based upon a practical geometry. The results obtained are promising and show that the code parallelizes well. The numerical approach leads to an attractive tool for treating real-life large-scale problems.

Future research will mainly address the optimization of the present method, further validation and new applications in the area of coastal transport. For example, we should like to include a more advanced domain decomposition method. The numbers found in Table 2 are still somewhat high. Moreover, we intend to study the mesh partitioning a little bit closer both for the present geometry and for more complicated geometries taken from coastal applications.

Acknowledgements

We like to thank Agip S.p.A. for providing field data. The research was carried out with financial support of the Dutch Ministery of Economic Affairs and the regional authorities of Sardegna. We also like to thank H.X. Lin (Delft University) for providing the data of the Chaco runs.

References

[1] L. Bergamaschi, S. Mantica, F. Saleri, Mixed finite element approximation of Darcy's law in porous media, Report CRS4 AppMath-94-20, CRS4, Cagliari, Italy, 1994.

[2] E. Brakkee, P. Wilders, The influence of interface conditions on convergence of Krylov–Schwarz domain decomposition for the advection–diffusion equation, Report 95-67, Delft Univ. of Techn., Fac. Techn. Math. and Inf., 1995.

[3] M.A. Driscoll, W.R. Daasch, Accurate predictions of parallel program execution time, J. Par. Distr. Comp. 25 (1995) 16–30.

[4] R.E. Ewing, The Mathematics of Reservoir Simulation, SIAM, Philadelphia, 1983.

[5] R.E. Ewing, T.F. Russell, Efficient time stepping procedures for miscible displacement problems in Porous Media, SIAM J. Numer. Anal. 19 (1982) 1–67.

[6] R.E. Ewing, T.F. Russel, M.F. Wheeler, Convergence analysis of an approximation of miscible displacement in porous media by mixed finite elements and a modified method of characteristics, Comput. Methods Appl. Mech. Eng. 47 (1984) 73–92.

[7] G. Fotia, A. Quarteroni, Modelling and simulation of fluid flow in complex porous media, in: K. Kirchgassner, O. Mahrenholtz, R. Mennicken (Eds.), Proc. ICIAM'95, Akademic Verlag, Berlin, 1996.

[8] J.L. Gustafson, G.R. Montry, R.E. Brenner, Development of a parallel method for a 1024-processor hypercube, SIAM J. Sci. Statist. Comput. 9 (1988) 532–533.

[9] B. Hendrickson, R. Leland, The Chaco user's guide, version 2.0, Report SAND 95-2344, Sandia National Lab., USA, 1995.

[10] A. Jameson, D. Mavripilis, Finite volume solution of the two-dimensional Euler equations on a regular triangular grid, AIAA J. 24 (1986) 611–618.

[11] A. Jameson, W. Schmidt, E. Turkel, Numerical solution of the Euler equations by finite volume methods using Runge–Kutta time-stepping schemes, AIAA Paper 81-1259, 1981.

[12] T. Kaarstad, J. Froyen, P. Bjørstad, M. Espedal, A massively parallel reservoir simulator, in: Proc. 13th Symp on Reservoir Simulation, TX, USA, Soc. Petr. Eng. Inc. (SPE29139), 1995.

[13] W. Kinzelbach, Groundwater Modelling, Elsevier, Amsterdam, 1986.
[14] L.W. Lake, Enhanced Oil Recovery, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[15] M.J. Quinn, Parallel Computing: Theory and Practice, 2nd ed., McGraw-Hill, New York.
[16] B.F. Smith, P.E. Bjørstad, W.D. Gropp, Domain Decomposition; Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, Cambridge, UK, 1996.
[17] L. Stolcis, L.J. Johnston, Solution of the Euler equations on unstructured grids for two-dimensional compressible flow, Aeronautical J. 94 (1990) 181–195.
[18] X.-H. Sun, L.M. Ni, Scalable problems and memory bounded speed-up, J. Par. Distr. Comp. 19 (1993) 27–37.
[19] G. Thomas, D. Thurnau, Reservoir simulation using an adaptive implicit method, SPEJ 23 (1983) 759–768.
[20] R.C. Wattenbarger, Simulation of tracer flow through heterogeneous porous media, Ph.D. thesis, Stanford University, Stanford, USA, 1992.
[21] R.C. Wattenbarger, K. Aziz, F.M. Orr, High-Troughput TVD-based simulation of tracer flow, in: Proc. 13th Symp. on Reservoir Simulation, TX, USA, Soc. Petr. Eng. Inc. (SPE29097), 1995.
[22] P. Wilders, E. Brakkee, Schwarz and Schur: a note on finite volume domain decomposition for advection–diffusion, Report 95-59, Delft Univ. of Techn., Fac. of Techn. Math. and Inf., 1995.
[23] P. Wilders, G. Fotia, M. Marrone, Implicit time stepping and domain decomposition for 2D miscible flow in porous media with unstructured finite volumes, Report CRS4 AppMath-94-23, CRS4, Cagliari, Italy, 1994.